

## LAYERED USER INTERFACE

## FIELD OF THE INVENTION

5 The present invention relates to user interfaces and in particular to the user interface for devices for use with a mobile communications network.

## BACKGROUND OF THE INVENTION

10 There is now a significant market in downloading images, ringtones, wallpapers, etc., to enable users to modify the appearance of their mobile phones. For commercial reasons it is desirable for mobile network operators and/or content providers to be able to have some control over the user interface that will be displayed on the screen of a mobile device. Conventional methods for implementing user interfaces lack the flexibility and configurability that enable such schemes to be implemented.

15

## SUMMARY OF THE INVENTION

According to a first aspect of the present invention, there is provided a method of generating a user interface for a device, the method comprising the steps of: (a) generating a plurality of sets of user interface elements, each of the plurality of sets of user interface elements comprising one or more user interface elements, wherein the or each user interface element is associated with a defined region of the user interface; (b) ordering each of the plurality of sets of user interface elements into an sequence; (c) querying each of the plurality of sets of user interface elements to

select a plurality of user interface elements for use in the user interface, the sets being queried in accordance with the ordering performed in step (b), wherein if more than one user interface element is associated with the same region of the user interface then the selected user interface element is taken from the set of elements which occurs first within the sequence determined in step (b); and (d) rendering the user interface in accordance with the plurality of user interface elements selected in step (c).

10

According to a second aspect of the present invention, there is provided a device comprising a display means and a user interface being displayed by the display means, the device being configured, in use, to: (a) generate a plurality of sets of user interface elements, each of the plurality of sets of user interface elements comprising one or more user interface elements, wherein the or each user interface element is associated with a defined region of the user interface; (b) order each of the plurality of sets of user interface elements into an sequence; (c) query each of the plurality of sets of user interface elements to select a plurality of user interface elements for use in the user interface, the sets being queried in accordance with the ordering performed in step (b), wherein if more than one user interface element is associated with the same region of the user interface then the selected user interface element is taken from the set of elements which occurs first within the sequence determined in step (b); and (d) render the user interface in accordance with the plurality of user interface elements selected in step (c).

According to a third aspect of the present invention, there

is provided a data carrier comprising computer executable code for performing any of the above-described methods.

#### BRIEF DESCRIPTION OF THE DRAWINGS

5

Figure 1 shows a schematic depiction of a system incorporating the present invention;

Figure 2 depicts in greater detail the structure and operation of server;

10 Figure 3 shows a schematic depiction of the software for the mobile devices;

Figure 4 shows a schematic depiction of four hierarchical planes; and

15 Figure 5 shows a schematic depiction of a device that comprises a user interface according to an embodiment of the present invention.

#### DETAILED DESCRIPTION OF THE INVENTION

20 The invention will now be described by way of illustration only and with respect to the accompanying drawings, in which Figure 1 shows a schematic depiction of a system incorporating the present invention. The system comprises server 100, content toolset 200, mobile devices 300, 25 operational support systems (OSSs) 700, content feeds 500 and user interface (UI) sources 600. In use, the server 100 communicates content data and UI data to the mobile devices 300, 301, ..., each of which comprise software package 400. The server 100 interfaces with OSSs 700, with the OSSs being those conventionally used to operate mobile networks, for example billing, account management, etc. The server 100 further interfaces with the content toolset 200: the content

toolset receives data from UI sources 600, 601, ..., and packages the UI data such that the server can transmit the packaged UI data to the software packages 400 comprised within the mobile devices 300. The server receives data from 5 a plurality of content feeds, and this data is processed and packaged such that it can be sent to the software packages 400 or so that the mobile devices 300 can access the data using the software package 400.

10 The system can be envisaged as being divided into three separate domains: operator domain 50 comprises the systems and equipment operated by the mobile network operator (MNO); user domain 60 comprises a plurality of mobile devices and third-party domain 70 comprises the content feeds and UI 15 feeds that may be controlled or operated by a number of different entities.

Figure 2 depicts in greater detail the structure and operation of server 100. Server 100 comprises publishing 20 component 110 and content server component 150. Publishing component comprises database 111, import queue 112, content toolset interface 113, user interface 114 & catalogue 115. In operation, the publishing component receives content from the content toolset at the content toolset interface. The 25 content is presented in the form of a parcel 210a, 210b, ..., (see below) comprising one or more Trigs and one or more Triglets. A trig is a user interface for a mobile device, such as a mobile telephone and a triglet is a data file that can be used to extend or alter a trig. If a parcel comprises 30 more than one trig then one of the Trigs may be a master trig from which the other Trigs are derived.

The publishing component user interface 114 can be used to import a parcel into the database 111, and this process causes references to each trig and triglet to be loaded into the import queue 114, which may comprise references to a 5 plurality of parcels 210a, 210b, ... . The contents of the parcel may be examined using the user interface and the contents of the parcel can be passed to the catalogue.

The MNO may have several publishing domains, for example one 10 for each target server in a number of countries or regions. Each domain is treated in isolation from other domains and has its own publishing scheme describing how objects are to be published onto content servers in both live and staging environments. The publishing component GUI provides several 15 different views to each domain, enabling operators to completely manage the publishing of content. The catalogue comprises references to the Trigs stored in the catalogue and the update channels and feed channels used to transfer content to the various domains. For each domain, the 20 operator uses the publishing component GUI to set up the domain structure and allocate trigs from the catalogue to each domain node. To aid the operator in selecting trigs efficiently, a filter is provided in the catalogue so that only relevant items are shown.

25 The content server component 150 is a standard implementation of a web server and as such the scaling model is well understood. The capabilities of a server can be rated using a "SPECweb99" number indicating the number of concurrent 30 sessions that the web server can handle under benchmark conditions. Published SPECweb99 numbers range from 404 to 21,000 with typical commercial web servers having SPECweb99

numbers in the order of 5,000. A typical deployment scenario of 1m subscribers with hourly updating content requires a web server with a SPECweb99 rating of only 1,112. A successful deployment will lead to increased service use which can be 5 provided for by enabling additional servers to create an infrastructure that can be both scalable and highly resilient to failure.

A connection may be made to the server from a mobile device 10 via a WAP Gateway. In this case the web server session exists between the WAP gateway and the web server, rather than the mobile phone and web server. When a request is made for a file via the WAP gateway, the session with the web server lasts only as long as it takes to transfer the file from the 15 web server to the WAP gateway - i.e. the session is extremely short since the connection bandwidth will be very high and latency extremely low.

Alternatively a direct connection may be established between 20 the mobile phone and the web server. In this case, the web server will need to keep the session open for as long as it takes to download the data to the phone.

There are two types of content that are delivered by the 25 content server component: trigs, typically of the order of 100KB and regularly updating triglets which are typically of the order of 1KB. The traffic created by trig downloads is very similar to the traffic generated by existing content downloads. And thus the related issues are well understood. 30 Downloads of regular triglet updates are a new feature in an MNO's traffic model but because of the small size of the updates, which typically fit within one data packet, it is

possible to show that the traffic can still be handled by typical web servers.

In the case of a triglet download, typically only one data 5 packet is required to transfer 1KB. Assuming a round-trip latency across a GPRS network of 2 seconds, the web server will need to hold open a typical session for around 4 seconds. For the scenario of 1 million subscribers having a trig on their phone with content that updates every hour, 10 this implies 278 hits per second on the web server and 1,112 concurrent sessions. As stated earlier, this number is well within the capability of typical web servers.

Figure 3 shows a schematic depiction of the software 400 for 15 the mobile devices 300, which comprises a mark-up language renderer 410, update manager 420, network communication agent 425, resource manager 430, virtual file system 435, actor manager 440, a plurality of actors 445a, 445, ..., native UI renderer 450, support manager 460, trig manager 465 and mark- 20 up language parser 470.

It is preferred that the software operates using TrigML, which is an XML application and that mark-up language renderer 410 renders the TrigXML code for display on the 25 mobile device 300. The mark-up language renderer also uses the TrigML Parser to parse TrigML resources, display content on the device screen and controlling the replacement and viewing of content on the handset. The native UI renderer is used to display UI components that can be displayed without 30 the use of TrigML, and for displaying error messages.

The software 400 is provisioned and installed in a device

specific manner. For example for a Nokia Series 60 device the software is installed using a SIS file, whereas for a MS Smartphone device the software is installed using a CAB file. Similarly, software upgrades are handled in a device specific 5 manner. The software may be provisioned in a more limited format, as a self-contained application that renders its built in content only: i.e. the software is provisioned with a built-in trig but additional trigs cannot be added later. The supplied trig may be upgraded over the air.

10

The trig manager 465 presents an interface to the resource manager 430 and the mark-up language renderer. It is responsible for trig management in general. This includes: persisting knowledge of the trig in use, changing the current 15 trig, selection of a trig on start-up, selection of a further trig as a fall back for a corrupt trig, maintaining the set of installed trigs, identifying where a particular trig is installed to the resource manager and reading the update channel definitions of a trig and configuring the update 20 manager appropriately.

The resource manager provides an abstraction of the persistent store on device, i.e. storing the files as real files, or as records in a database. The resource manager 25 presents a file system interface to the mark-up language renderer and the update manager. It is responsible for handling file path logic, distinguishing between real resource files and actor attributes, mapping trig-relative paths onto absolute paths, interfacing with the trig manager 30 and providing a modification interface to the update manager.

The Update Manager handles the reception and application of

Trigs and Triglets. The Update Manager presents an interface to the Renderer and the trig Manager and is responsible for: the initiation of manual updates when instructed to by the Renderer; controlling and implementing the automatic update 5 channel when so configured by the trig manager; indicating the progress of a manual update and recovering an Update following unexpected loss of network connection and/or device power. The update packet format may be defined as a binary serialisation of an XML schema.

10 The Support Manager provides an interface for other components to report the occurrence of an event or error. Depending on the severity of the error, the Support Manager will log the event and/or put up an error message popup.

15 XML is a convenient data formatting language that is used to define the update packet format as well as TrigML content. For bandwidth and storage efficiency reasons, text XML is serialised into a binary representation. Both update packets 20 and TrigML fragments are parsed by the same component, the mark-up language parser. Any further use of XML in the software must use the binary XML encoding and therefore re-use the parser.

25 The Actor Manager 440 looks after the set of actors 445 present in the software. It is used by: the renderer when content is sending events to an actor; actors that want to notify that an attribute value has changed and actors that want to emit an event (see below).

30 The software may comprise a multi-threaded application running a minimum of two threads, with more possible

depending on how many and what sort of actors are included. The software runs mostly in one thread, referred to as the main thread. The main thread is used to run the renderer which communicates synchronously with other components.

5 Actors always have a synchronous interface to the Renderer. If an actor requires additional threads for its functionality, then it is the responsibility of the Actor to manage the inter-thread communication. It is preferred that a light messaging framework is used to avoid unnecessary code

10 duplication where many actors require inter-thread communication.

In addition to the main thread, the update manager runs a network thread. The network thread is used to download

15 update packets and is separate from the main thread to allow the renderer to continue unaffected until the packet has arrived. The Update Manager is responsible for handling inter-thread messaging such that the Update Manager communicates synchronously with the Renderer and Resource

20 Manager when applying the changes defined in an Update Packet.

The Renderer receives information regarding the key press. If there is no behaviour configured at build time for a key, it

25 is sent as a TrigML content event to the current focus element. The content event is then handled as defined by TrigML's normal event processing logic.

For example, if a key is pressed down, a 'keypress' event is

30 delivered to the Renderer with a parameter set to the relevant key. When the key is released, a '!keypress' event is delivered to the Renderer. If a key is held down for a

extended period of time, a 'longkeypress' event is delivered to the renderer. On release, both a '!longkeypress' and a '!keypress' event are delivered to the Renderer.

5

A trig is started by loading the defined resource name, start-up/default. The TrigML defined in start-up/default is parsed as the new contents for the content root node.

10 The first time a trig is run by the software following its installation, the trig is started by loading the resource name startup/firsttime. The software may record whether a trig has been run or not in a file located in the top level folder for that trig. Dependent on the platform used by the  
15 mobile device, the automatic start-up of the software may be set as a build-time configuration option. Furthermore, placing the software in the background following an auto-start may also be a build-time configuration option.

20 A launcher may appear to the user as an application icon and selecting it starts the software with a trig specified by that launcher (this trig may be indicated by a launcher icon and/or name). When using a launcher to start a trig, it is possible to specify an 'entry point' parameter. The parameter  
25 is a resource name of a file found in the 'start-up' folder. This file is not used if the trig has never been run before, in which case the file called 'firsttime' is used instead.

30 The software uses content resource files stored in a virtual file system on the device. The file system is described as virtual as it may not be implemented as a classical file-system, however, all references to resources are file paths

as if stored in a hierarchical system of folders and files. Furthermore, the software stores some or all of the following information: usage statistics; active user counts; TrigManager state; TrigML fragments & update channel 5 definition (serialised as binary XML); PNG images; plain text, encoded as UTF-8 OTA and then stored in a platform specific encoding; other platform specific resources, e.g. ring tone files, background images, etc.

10 Files in the file system can be changed, either when an actor attribute value changes, or when a file is replaced by a triglet. When files in the /attrs directory change, the Renderer is immediately notified and the relevant branches of the content tree are updated and refreshed. When images and 15 text resources are changed, the Renderer behaves as if the affected resources are immediately reloaded (either the whole content tree or just the affected branches may be refreshed). When TrigML fragments are changed, the Renderer behaves as if it is not notified and continues to display its current, 20 possibly out of date, content. This is to avoid the software needing to persist <include> elements and the <load> history of the current content.

25 The software 400 is provisioned to mobile devices in a device specific method. One or more Trigs can be provisioned as part of the installation, for example, stored as an uncompressed update packet. On start-up, the packet can be expanded and installed to the file-system.

30 The Actors 445 are components that publish attribute values and handle and emit events. Actors communicate with the Renderer synchronously. If an actor needs asynchronous

behaviour, then it is the responsibility of the actor to manage and communicate with a thread external to the main thread of the Renderer. An Actor can be messaged by sending it an event with the `<throw>` element. Events emitted by 5 actors can be delivered to the content tree as content events: these can be targeted at an element Id or 'top'. The interface to an actor is defined by an Actor Interface Definition file. This is an XML document that defines the attributes, types, fieldnames, events-in and parameters, and 10 events out. The set of actors is configurable at build-time for the software.

For commercial reasons it is desirable for MNOs and/or content providers to be able to have some control over the 15 user interface that will be displayed on the screen of a mobile device. It is also important that there is a degree of flexibility that allows users to download triglets or new trigs to modify the appearance of their device and also to make further changes to the displayed image that is 20 determined by the trig or triglet in use.

Content for the user interface is stored in archive files and the UI of an app can be defined by one or more archive files. Each archive file may contain multiple resources (mark-up 25 elements, images, text etc). The files within the archive are, like other archive file formats, stored in a tree structure, in a manner similar to a known folder/file structure. Where more than one archive file is used to define a UI, the required archive files are arranged in a 30 strict sequence (or list). When a resource is required it is searched for in each archive file, the search returning the first file found.

It is possible to order the archive files in such a manner that resources extracted from an archive file earlier in the list mask resources found in an archive file (or files) which 5 are found lower down in the list of archive files. For example, if there is a requirement to temporarily obscure an element of the UI, for example a window, then a mask element can be defined in an archive file which is stored at a higher level than the archive file that comprises the UI element to 10 be obscured.

It should be noted that when the user interface content is interpreted to render the UI, the addition of an obscuring element causes the obscuring element to be rendered within 15 the UI. As the obscuring element and the UI element to be masked occupy a common region of the UI and the obscuring element is held within an archive file which is a higher layer than the archive file that comprises the element to be masked, only the obscuring element is to be shown in the UI 20 and thus only the obscuring element is fetched by the renderer in order to render the UI. Because the obscuring element is to be rendered and it occupies the same position as the element to be masked, the element to be masked is not fetched by the renderer. This approach reduces the amount of 25 data that is fetched by the renderer in order to render the UI and thus reduces the amount of processing power that is required to render the UI and decreases the time that is taken to render the UI.

30 This method is of significant use where an archive file is needed by more than one application on a device (such an archive file may be referred to as the common archive or base

archive). Each application can then supply their own application-specific archive file that masks some of the resources in the common archive file and adds further resources that are required. Each application has a list of 5 archive file(s) that it uses, but some of these archive files will be used by other applications as well. This can be thought of as a hierarchy of archive files: the common archive files are at the top layer of the hierarchy, with the application-specific archive files comprising the lower 10 layers of the hierarchy. Archive files specific to an application may be referred to as flat files, as the different archive files are to be found in the same layer of the hierarchy.

15 According to a further embodiment the above arrangement may be extended to allow the configuration and appearance of the UI to be controlled. For example, as the UI is effectively formed from the interaction of the elements that are stored in archive files that are associated with different layers of 20 the hierarchy, it is possible to assign one or more layers of the hierarchy to different entities, such as the MNO, device manufacturer, trig provider, the device user, etc. This allocation of the different layers of the hierarchy, for example, enables a logo associated with the device 25 manufacturer to be permanently displayed within the UI by assigning the top layer of the hierarchy to the manufacturer. If the network operator, or a trig provider, wishes to publicise a time-limited promotional offer then a suitable element can be added to an archive file in a layer associated 30 with the network operator. The entity that is allocated the lowest level(s) of the hierarchy is able to change and modify all of the UI elements that have not been defined in archive

files in any of the higher levels of the hierarchy.

Figure 4 shows a schematic depiction of four hierarchical planes 405a-d: plane 405a comprises UI elements defined by the MNO; plane 405b comprises UI elements defined by the device manufacturer; plane 405c comprises UI elements defined by a trig; and plane 405d comprises UI elements defined by the user. Plane 405a has the highest position in the hierarchy and plane 405d has the lowest position in the hierarchy. For example, the `mno_logo` element in plane 405a defines the graphic element used and its position on the display screen of the device. As it is in the highest plane of the hierarchy it will always appear and will take preference over any other UI element in a lower hierarchy element that attempts to use the pixels used by `mno_logo`. Plane 405d comprises the `backgroundcolour` element, which is not defined in any of the other planes and thus the colour defined in `backgroundcolour` will be used in the UI.

Plane 405c comprises the `windowtitle.txt` element that defines the attributes for the text used in the title of a window. This may be overwritten by adding a `windowtitle.txt` element to either plane 405a or 405b to define the text attributes, or by adding a `windowtitle.txt_deleted` element to either plane 405a or 405b to instruct the UI renderer to ignore any subsequent `windowtitle.txt` element.

This enables each feature of the UI to be configurable but also provides a framework within which certain UI elements can be defined by an entity, such as the network operator, in a manner that prevents other entities from altering or interfering with those defined UI element(s).

Although the preceding discussion has referred to user interface content being stored within archive files, it will be readily understood that the user interface content may 5 alternatively be stored in other forms that provide a collection of files, for example, a folder of unpacked files, or a folder comprising unpacked files and other folders.

10 In conventional mobile devices, information regarding the battery strength, signal strength, new text messages, etc. are shown to the user. Typically this information is obtained by the operating system sending a call to the relevant hardware or software entity and the UI interpreting 15 the received answer and displaying it.

This information may be displayed in the UI using a TrigML tag (see below) <phonestatus> (or <signalstrength>). Rendering this tag causes a listening query to be opened to 20 the relevant hardware or software entity. If a change in state occurs then the UI renderer is notified and the UI renderer loads the relevant icon or graphic to communicate the change in state to the user. If the user changes the view within the UI the tag may be withdrawn and the 25 listening query is terminated. This approach is more resource efficient as the listening query is only active when the tag is in use.

TrigML can use constant variables instead of attribute 30 values. Constant variables are accessed with the same syntax as <include> parameters, e.g. \$background\_colour. Constants are treated as global variables in a trig and are defined in

the reserved folder, constants/. The variable definitions contained in the files in the constants/ folder may be resolved at compile time with direct substitution of their values. In an alternative embodiment the variable 5 definitions in constants/ are compiled as global variables and resolved at content parse time by the software. This allows the trig to be updated by a simple replacement of one or all of its constants files.

10 A System String Dictionary defines the integer values to use for all well known strings, i.e. reserved words. These have several types, including: TrigML element and attribute names ('group', 'res', 'layer', 'image', 'x'), TrigML attribute values (e.g.: 'left', 'activate', 'focus') and common 15 resource paths (e.g.: 'attr', 'start-up', 'default'). As an input, the String Dictionary is optional. The first time a trig is compiled it will not have a String Dictionary. This first compilation creates the String Dictionary, which is then used for all future compilations of that trig. Triglet 20 compilation must have a String Dictionary that defines all the string mappings used by the trig it is modifying.

In order to successfully render the user interface of a mobile device, the mark-up language must have the following 25 qualities: concise page definitions, consistent layout rules, be implementable in a compact renderer, provide multiple layering and arbitrary overlapping content, event model, require the repaint of only the areas of the display that have to change between pages of the UI, include hooks to the 30 platform for reading property values receiving events and sending events, extensible, and be graphically flexible. TrigML provides these features and an overview of the

elements and attributes that provide the desired functionality can be found in our co-pending application GB0403709.9, filed February 19th 2004.

5 It is desirable that the cost of re-branding UIs and producing a continual stream of updates is minimal. This is enabled by providing an efficient flow of information from the creative process through to the transmission of data to users.

10 A container, referred to as a parcel, is used for UIs, UI updates, and templates for 3rd party involvement. Parcels contain all the information necessary for a 3rd party to produce, test and deliver branded UIs and updates. Figure 5  
15 shows a schematic depiction of the content toolset 200, which comprises scripting environment 220, test and simulation environment 230 and maintenance environment 240

The parcel process comprise five processing stages:

20 1) The scripting environment 220 provides the means to design the template for one or more UIs and the update strategy for UIs based on that template.

2) The maintenance environment 240 provides for rapid UI and update production in a well-controlled and guided environment  
25 that can be outsourced to content providers.

3) The maintenance environment 240 'pre-flight' functionality allows the deployment administrator to check and tune the UIs and updates that they receive from 3rd parties.

4) The publishing component 110 provides management of UIs  
30 and updates at the deployment point, including the staging of new releases.

5) The publishing component 110 enables the automatic

generation of updates from live content feeds.

In a typical project, parcels are created within the scripting environment 220 for: a content provider to create 5 re-branded UIs from a template, incorporating the same 'feel' but a different 'look'; a content provider to create updates from a template, that provide a periodic, or user selected variation to UI content ; or an ad agency to create updates from a template that promote new services on a periodic 10 basis.

For all of these use cases, maintenance environment 240 is used to import the parcel, re-brand and reconfigure the content and create a new parcel for submission to the 15 publishing component 110. In the design of the UI template, the following issues should be considered: which part of the UI can be re-branded; which features of a UI need to be reconfigured at re-branding or remotely; which part of the UI content may be updated; and if the UI is re-branded then can 20 user select content feeds in use. The scripting environment 220 allows these strategies to be defined, and enables the maintenance environment 240 as the implementer of each instance of each strategy.

25 A parcel is generated by the scripting environment 220 which comprises a template UI or update for editing. Once editing is complete the parcel is saved in an 'outbox' ready for despatch to the maintenance environment 240 for publishing to the content server. The following 'parcel' functions are 30 provided. The maintenance environment 240 can be used to edit/replace resources held within the parcel. Parcels can be exported to the simulation environment to test the

performance of the UI or UI update on a mobile device.

A parcel entry may be double clicked to launch an appropriate editor. (for example, an image resource would launch an image editor). All resources may have a text description/note inserted in the maintenance environment and displayed in the appropriate context in the maintenance environment. Lists of menu entries are handled as a special resource type with each entry presenting its own sub-catalogue of resources (for example - title, help string, image, roll-over image, URL and ringtone preview).

Many different UIs can be derived from a common base. Typically the common base would implement most of the interface itself, and Trigs derived from it would implement small variations on it, such as branding. A Triglet can be derived from a Trig, and it can override any of the resources from the parent Trig that it chooses to (optionally it may introduce its own resources). Note that "resources" here also refers to TrigML, so the behaviour and layout of a Trig can be modified by a Triglet just as easily as it replacing a single image or piece of text.

A Parcel may comprise one or more base Trigs (i.e. a Trig that is not derived from any other trig), one or more multiple Trigs derived from a base Trig, a plurality of triglets derived from any of the trigs and a plurality of triglets derived from other triglets.

Figure 5 shows a schematic depiction of a device 800 that comprises a user interface according to an embodiment of the present invention. The device comprises a display 810 that

displays the user interface 815 and user interface means 820, that enable the user to interact with the user interface 815. A processor 830 executes the software that is stored within one or more storage means 840 and there may be provided one 5 or more wireless communication interfaces 850, to enable communication with other devices and/or communication networks. One or more batteries 860 may be received to power the device, which may also comprise interfaces to receive electrical power and/or communication cables.

10

The nature of these components and interfaces will depend upon the nature of the device. It will be understood that such a user interface can be implemented within a mobile or cellular telephone handset, but it is also applicable to 15 other portable devices such as digital cameras, personal digital organisers, digital music players, GPS navigators, portable gaming consoles, etc. Furthermore, it is also applicable to other devices that comprise a user interface, such as laptop or desktop computers.

20

The user interface means may comprise a plurality of buttons, such as a numerical or alpha-numerical keyboard, or a touch screen or similar. One or more storage devices may comprise a form of non-volatile memory, such as a memory card, so that 25 the stored data is not lost if power is lost. ROM storage means may be provided to store data which does not need updating or changing. Some RAM may be provided for temporary storage as the faster response times support the caching of frequently accessed data. The device may also accept user 30 removable memory cards and optionally hard disk drives may be used as a storage means. The storage means used will be

determined by balancing the different requirements of device size, power consumption, the volume of storage required, etc.

Such a device may be implemented in conjunction with 5 virtually any wireless communications network, for example second generation digital mobile telephone networks (i.e. GSM, D-AMPS), so-called 2.5G networks (i.e. GPRS, HSCSD, EDGE), third generation WCDMA or CDMA-2000 networks and improvements to and derivatives of these and similar 10 networks. Within buildings and campuses other technologies such as Bluetooth, IrDa or wireless LANs (whether based on radio or optical systems) may also be used. USB and/or FireWire connectivity may be supplied for data synchronisation with other devices and/or for battery 15 charging.

Computer software for implementing the methods and/or for configuring a device as described above may be provided on data carriers such as floppy disks, CD-ROMS, DVDs, non- 20 volatile memory cards, etc.

This application claims the benefit of UK Patent Application number 0403709.9, filed February 19th 2004, the contents of which are incorporated herein by reference.